Modul Pelatihan Teaching Factory Programming: Navigasi dan Lokalisasi dengan RTAB-Map

Program pelatihan ini dirancang untuk mahasiswa yang sudah memiliki pengetahuan dasar tentang ROS2. Fokus pelatihan adalah pada navigasi, lokalisasi, navigasi waypoint, dan pengendalian misi menggunakan state machine. Pelatihan ini menggunakan simulasi TurtleBot dan RTAB-Map.

1 Modul 1: Setup & Installasi

1.1 Konfigurasi Environment

- Install ROS 2 Humble lakukan installasi menggunakan **panduan instalasi Debian**
- Install TurtleBot3 packages menggunakan **panduan instalasi TurtleBot3**
- Install Gazebo dengan `sudo apt install ros-humble-desktop`
- Install RViz2 dengan `sudo apt install ros-humble-rviz2`
- Install RTAB-Map dengan `sudo apt install ros-humble-rtabmap-ros`
- Install Navigation2 dengan `sudo apt install ros-humble-navigation2 ros-humble-nav2-bringup`

1.2 Menjalankan Simulasi

Setelah semua paket terinstal, jalankan simulasi TurtleBot3 Waffle Pi di Gazebo dengan environment `house`:

```
export TURTLEBOT3_MODEL=waffle_pi
ros2 launch turtlebot3_gazebo turtlebot3_house.launch.py
```

1.3 Teleoperation

Buka terminal lain untuk menjalankan teleop:

```
ros2 run turtlebot3_teleop teleop_keyboard
```

1.4 RViz2

Buka RViz2 untuk melihat topik yang dipublish dan TF yang dimiliki robot:

```
rviz2
```

Atur Fixed Frame ke `odom`. Tambahkan display untuk `LaserScan`, dan `TF`. Sehingga akan muncul tampilan seperti berikut:

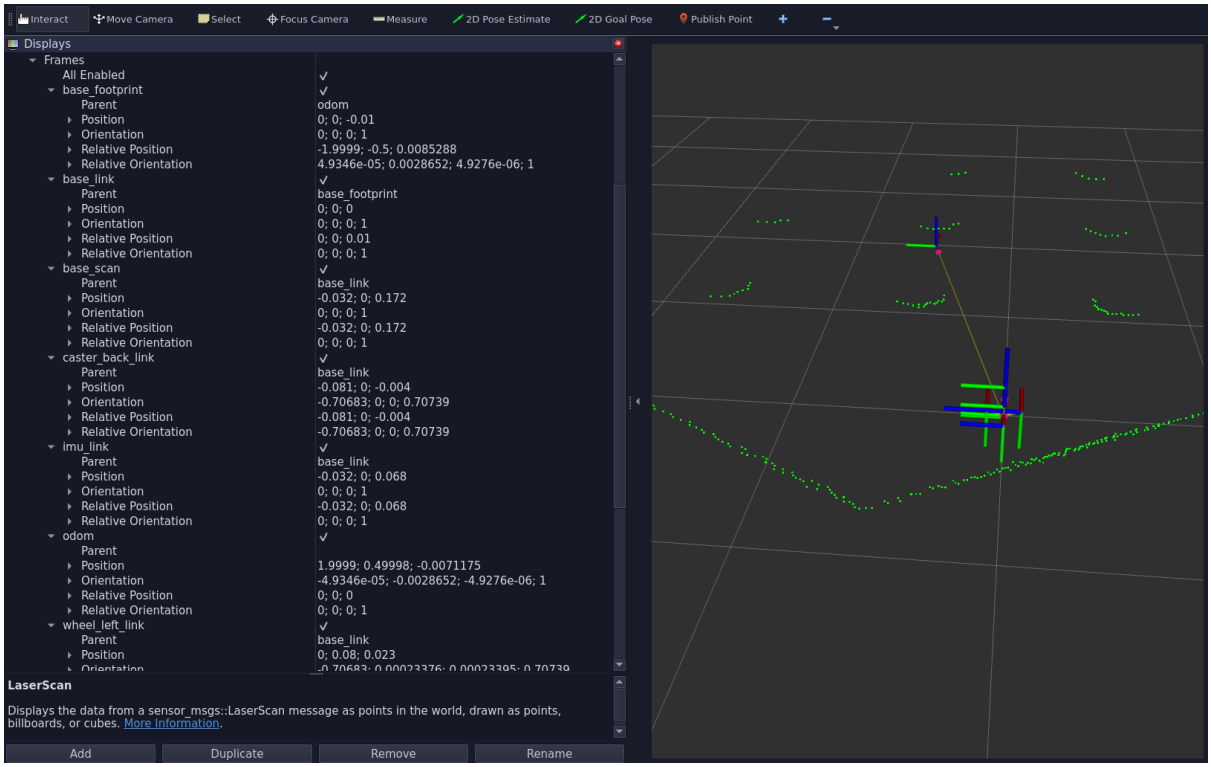

Figure 1: Contoh tampilan RViz2

Pada sidebar ditunjukkan TF yang dimiliki robot:

- `odom`: frame odometri, bergerak relatif terhadap `base_link`
- `base_footprint`: frame footprint robot (sejajar dengan ground plane 2D)
- `base_link`: frame utama robot (dapat mengalami translasi dan orientasi)
- `caster_back_link`: frame caster back (free wheel)
- `imu_link`: frame IMU
- `base_scan`: frame laser scan
- `wheel_left_link`: frame roda kiri
- `wheel_right_link`: frame roda kanan
- `camera_link`: frame dasar kamera (body), menunjukkan posisi fisik kamera pada robot. Orientasi mengikuti konvensi ROS, yaitu x ke depan, y ke kiri, dan z ke atas.
- `camera_rgb_frame`: frame kamera RGB (sensor), digunakan sebagai acuan untuk data gambar. Orientasinya sama dengan `camera_link`.
- `camera_rgb_optical_frame`: frame kamera dengan konvensi optik. Orientasi mengikuti standar OpenCV, yaitu z ke depan, x ke kanan, dan y ke bawah.

Semua TF tersebut terhubung dalam sebuah pohon (tree) yang dapat dilihat pada tab `TF` di RViz2. atau dengan perintah:

```
ros2 run tf2_tools view_frames.py
```

Perintah ini akan menghasilkan file `frames.pdf` yang menunjukkan struktur pohon TF sebagai berikut:

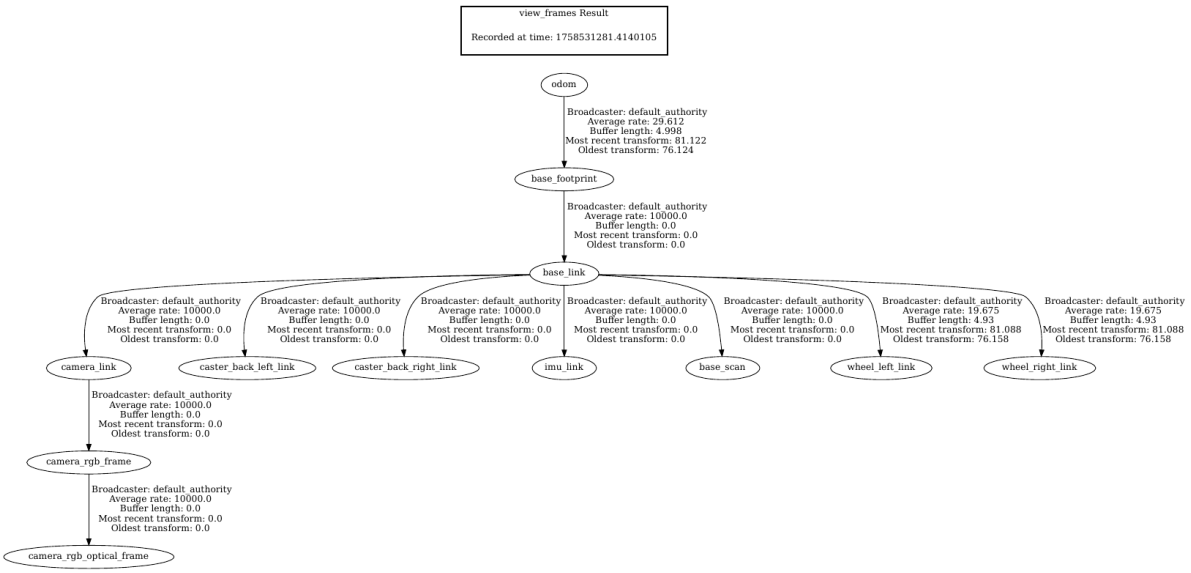

Figure 2: Contoh TF Tree

2 Modul 2: Mapping dengan RTAB-Map

Objectives:

2.1 SLAM dan RTAB-Map

SLAM (Simultaneous Localization and Mapping) adalah proses di mana robot membangun peta lingkungan sambil menentukan posisinya di dalam peta tersebut. Pada dasarnya, SLAM dibagi menjadi dua yaitu map based SLAM dan graph based SLAM. Perbedaan paling mendasar adalah pada bagian lokalisasinya. Pada map based SLAM, lokalisasi dilakukan dengan melakukan registrasi data sensor ke map yang telah dibangun sebelumnya. Sedangkan pada graph based SLAM, lokalisasi dilakukan dengan registrasi sensor ke setiap graph yang dibangun sebelumnya.

RTAB-Map (Real-Time Appearance-Based Mapping) adalah sebuah algoritma graph based SLAM yang menggunakan data visual (gambar) dan data jarak (misalnya dari LIDAR atau depth camera) untuk membangun peta 3D dari lingkungan sekitar. RTAB-Map dapat digunakan untuk berbagai aplikasi robotika, termasuk navigasi, inspeksi, dan pemetaan lingkungan. Untuk memperjelas bagaimana cara kerja Graph Based SLAM, silahkan simak video berikut: **Graph Based SLAM Explanation**.

2.2 RTAB-Map

Kunci dari Graph Based SLAM adalah deteksi loop closure. Di RTAB-Map, deteksi loop closure menggunakan kamera. Algoritma dasar yang perlu dipelajari untuk mencari loop closure menggunakan kamera adalah Image Registration. Image Registration adalah proses mencocokkan dua gambar yang diambil dari sudut pandang yang berbeda untuk menemukan transformasi geometris antara keduanya. Contoh contoh algoritma yang sering dipakai adalah GFTT, FAST, BRIEF, ORB, SIFT, dan lain lain.
RTAB-Map bukanlah Ready to Use SLAM, perlu dilakukan penyesuaian pada robot yang akan digunakan!

2.3 Penyesuaian RTAB-Map

2.3.1 Penyesuaian Sensor

Di RTAB-Map, bisa menggunakan beberapa sensor, cara setting nya lewat parameter:

- **subscribe_depth**: Untuk kamera depth seperti realsense
- **subscribe_scan**: Untuk LIDAR 2d

- **subscribe_scan_cloud**: Untuk LIDAR 3d
- **subscribe_rgbd**: Untuk kamera RGBD seperti Kinect
- **subscribe_stereo**: Untuk kamera stereo seperti Zed
- **subscribe_odom**: Untuk odometri hardware atau sumber odometry lainnya

Jika sistem robot tidak memiliki source odometry, odometry bisa dihitung dari visual odometry atau icp odometry bawaan RTAB-Map. Untuk menggunakannya bisa menajalankan node lain yaitu `rgbd_odometry` atau *icp_odometry*.

2.3.2 Penyesuaian Loop Closure Detection

Ada beberapa hal penting yang perlu diperhatikan dalam penyesuaian loop closure detection:

- **Kp/DetectorStrategy**: Memilih algoritma image matching yang sesuai dengan kondisi lingkungan dan jenis kamera yang digunakan.
- **Kp/MaxFeatures**: Berapa banyak maksimal fitur (words) yang bisa disimpan dalam database. Perhatikan juga setiap algoritma akan menghasilkan jumlah fitur yang berbeda beda.
- **Vis/MinInliers**: Hasil dari scan matching juga berupa inliers yang telah diperoleh dari RANSAC. Semakin besar nilai inliers, semakin baik hasil scan matching. Namun, jika terlalu besar, bisa jadi tidak ada loop closure yang terdeteksi.

2.3.3 Penyesuaian Graph Optimization

Graph optimization adalah proses memperbaiki posisi node pada graph berdasarkan loop closure yang terdeteksi. Ada beberapa parameter penting yang perlu diperhatikan:

- **Reg/Strategy**: Memilih mau menggunakan sensor apa untuk menghitung constraint per node pada graphnya. Bisa menggunakan icp atau visual atau gabungan keduanya. ICP dari LIDAR sedangkan visual dari kamera.
- **Optimizer/Strategy**: Memilih algoritma optimizer yang diinginkan (default GTSAM).
- **Optimizer/Iterations**: Semakin besar maka semakin lama iterasi optimasinya, hasilnya belum tentu lebih baik. Gunakan default saja jika tidak ada masalah.
- **RGBD/OptimizeMaxError**: Gerbang terakhir untuk optimasi, Jika hasil error nya terlalu besar maka loop closure akan di-reject dan graph tidak akan dioptimasi.
- **RGBD/OptimizeFromGraphEnd**: Ketika loop closure diterima, ada dua pendekatan optimasi, optimasi seluruh graph atau hanya dari node terakhir. Jika optimiasi dari seluruh graph, pasti akan membuat map atau estimasi pose robot loncat.

2.3.4 Parameter Penting Lainnya

- **Mem/IncrementalMemory**: Dipastikan false, agar database dalam keadaan frozen saat pertama kali dihidupkan. Intinya, keadaan default adalah keadaan lokalisasi, bukan mapping. Jika ingin mapping, harus diaktifkan dengan service.
- **Mem/STMSize**: Berapa banyak node terakhir yang akan disimpan di Short Term Memory (STM). Node-node ini akan dibandingkan dengan node baru untuk mencari loop closure. Jika robot bergerak cepat, maka STM size harus lebih besar.
- **Mem/NotLinkedNodesKept**: Berapa banyak node yang tidak terhubung dengan loop closure yang akan disimpan di memori. Setting ini ke True agar databse tidak crash.
- **RGBD/NeighborLinkRefining**: Jika TF sensor sudah dipastikan bagus, maka parameter ini bisa diset True. Jika kurang percaya dengan hasil sensor, dan ingin lebih percaya dengan odometry, maka bisa diset False.

2.3.5 Grid Map

Grid hasil dari RTAB-Map berupa occupancy grid. Hasil dari occupancy grid ini tidak penting untuk lokalisasi karena RTAB-Map merupakan graph based SLAM. Biasanya, grid map ini digunakan untuk navigasi yang nantinya berupa global costmap. Ada beberapa parameter penting yang perlu diperhatikan:

- **Grid/Sensor**: Jenis sensor yang digunakan untuk membuat grid. Bisa menggunakan laser scan, depth image, atau point cloud. Pilih sesuai dengan sensor yang dimiliki robot.
- **Grid/RangeMax**: Jarak maksimal dari sensor yang akan dimasukkan ke dalam grid. Jika jarak sensor lebih dari nilai ini, maka data tersebut akan diabaikan.
- **Grid/CellSize**: Ukuran sel grid. Semakin kecil ukuran sel, semakin detail peta yang dihasilkan, namun juga semakin besar ukuran peta dan semakin lama waktu pemrosesan.

3 Module 4: Navigation with Nav2

Objectives:

- Learn Navigation2 stack
- Configure global and local planners
- Tune costmap parameters

Exercises:

- Launch nav2_bringup with TurtleBot
- Send a navigation goal in RViz
- Observe path planning and execution

Final Project

Students implement a complete mission-based program:

- Explore and build map with RTAB-Map
- Localize using saved map
- Patrol using waypoints
- Implement mission logic with state machine